1

WRDC-TR-90-8007
Volume V
Part 35

WRIGHT RESEARCH & DEVELOPMENT CENTER

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 35 - DDL to NDDL Translator Development Specification

M. Apicella, S. Singh

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

David Judson

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

Bruce Rasmussen

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for Public Release;
Distribution is Unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
DS 620341410

5. MONITORING ORGANIZATION REPORT NUMBER(S)
WRDC-TR-90-8007 Vol. V, Part 35

6a. NAME OF PERFORMING ORGANIZATION
Control Data Corporation;
Integration Technology Services

6b. OFFICE SYMBOL *(if applicable)*

7a. NAME OF MONITORING ORGANIZATION
WRDC/MTI

6c. ADDRESS *(City,State, and ZIP Code)*
2970 Presidential Drive
Fairborn, OH 45324-6209

7b. ADDRESS *(City, State, and ZIP Code)*
WPAFB, OH 45433-6533

8a. NAME OF FUNDING/SPONSORING ORGANIZATION
Wright Research and Development Center,
Air Force Systems Command, USAF

8b. OFFICE SYMBOL *(if applicable)*
WRDC/MTI

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM.
F33600-87-C-0464

8c. ADDRESS *(City, State, and ZIP Code)*
Wright-Patterson AFB, Ohio 45433-6533

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
78011F	595600	F95600	20950607

11. TITLE *(Include Securitv Classification)*
[See block 19 ation

12. PERSONAL AUTHOR(S)
Control Data Corporation: Apicella, M. L., Singh, S.

13a. TYPE OF REPORT
Final Report

13b. TIME COVERED
4/1/87-12/31/90

14. DATE OF REPORT *(Yr.,Mo.,Day)*
1990 September 30

15. PAGE COUNT
22

16. SUPPLEMENTARY NOTATION
WRDC/MTI Project Priority 6203

17. COSATI CODES

FIELD	GROUP	SUB GR.
1308	0905	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify block no.)*

19. ABSTRACT *(Continue on reverse if necessary and identify block number)*

This development specification estabblishes the development, test, and qualification requirements of the DDL to NDDL translator computer program.

BLOCK 11:

INTEGRATED INFORMATION SUPPORT SYSTEM
Vol V - Common Data Model Subsystem

Part 35 - DDL to NDDL Translator Development Specification

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
David L. Judson

22b. TELEPHONE NO. *(Include Area Code)*
(513) 255-7371

22c. OFFICE SYMBOL
WRDC/MTI

EDITION OF 1 JAN 73 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.

Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface,and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

List of Figures

Table of Contents

Appendices

SECTION 1

SCOPE

1.1 Identification

This specification establishes the performance, development, test and qualification requirements of a computer program identified as the DDL to NDDL Translator. The DDL to NDDL Translator is one configuration item of the Integrated Information Support System (IISS) Common Data Model (CDM).

1.2 Functional Summary

This Computer Program Configuration Item (CPCI) is used to translate the Data Definition Language of a database management system to the Neutral Data Definition Language.

SECTION 2

DOCUMENTS

2.1 Reference Documents

[1] Systran, ICAM Documentation Standards, IDS 150120000C, 15 September 1983.

[2] D. Appleton Company, CDM Administrator's Manual, UM 620341000, 31 May 1988.

[3] D. Appleton Company, CDM1, An IDEF1 Model of the Common Data Model, CCS 620341000, 31 May 1988.

[4] Control Data Corporation, Neutral Data Definition Language User's Guide, 31 May 1988.

[5] C. J. Date, An Introduction to Database Systems, 1977, Addison-Wesley Publishing Company, Inc.

[6] IBM, DATABASE 2 Reference release 1.0, December 1984, IBM.

[7] Cincom Systems, TOTAL Database Administration Reference Manual, release 8.1 1978, Cincom Systems.

2.2 Terms and Abbreviations

Application Process: (AP), a cohesive unit of software that can be initiated as a unit to perform some function or functions.

Common Data Model: (CDM), IISS subsystem that describes common data of an enterprise and includes conceptual, external and internal schemas and schema transformations.

Common Data Model Administrator: (CDMA), the person or group of persons responsible for creating and maintaining an enterprises's Common Data Model. The CDMA manages the common data rather than managing applications that access data.

Common Data Model Processor: (CDMP), a component of the Common Data Model subsystem which is the distributed database manager of the IISS.

Conceptual Schema: (CS), the standard definition used for all data in the CDM. It is based on IDEF1 information modelling.

External Schema: (ES), an application's view of the CDM's conceptual schema.

Integrated Information Support System: (IISS), a computing environment used to investigate, demonstrate, test the concepts and produce application for information management and information integration in the context of Aerospace

Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Internal Schema: (IS), the definition of the internal model, the storage structure definition, which specifies how the physical data are stored and how they can be accessed. It is represented in terms of the physical database components, including record types and inter-record relationships.

Neutral Data Definition Language: (NDDL), A language used to manipulate and populate information in the Common Data Model (CDM) or IISS System Database.

Neutral Data Manipulation Language: (NDML), A language developed by the IISS project to provide uniform access to common data, regardless of database manager or distribution criteria. It provides distributed retrieval, single node update, and non-guaranteed distributed update.

Presentation Schema: (PS), The totality of the form fields in an application which are targets of data derivative from the common data.

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

This system is to be used to accomplish the translation of the native Data Definition Language (DDL) for DATABASE 2 and TOTAL data base management systems into the IISS Neutral Data Definition Language (NDDL) for the creation of the internal schema (IS) specification for the Common Data Model (CDM). The translator consists of several translators, one for each DDL which must be translated to NDDL. Each translator differs only in its lexical anaylzer and parser and to avoid confusion this document will refer to these translators collectively as "the translator".

3.1.1 System Capacities

The system capacities of the DDL to NDDL Translator have not been determined.

3.1.2 Interface Requirements

The input to the translator is constrained to agree with the DDL of the languages DATABASE 2 (SQL) and TOTAL DDL. The output of the translator is constrained to agree with the NDDL for specification of an IS to the CDM.

3.1.2.1 Interface Block Diagram

```
TOTAL DDL -->                  +------------------------+
DATABASE 2 SQL -->             | DDL to NDDL Translator |
                               +------------------------+


NDDL -->                       +----------------+
                               | NDDL Processor |
                               +----------------+
```

Figure 3-1 CDM IS Definitions

3.1.2.2 Detailed Interface Definition

The translator will take as input the DDL of either DATABASE 2 or TOTAL. The input is restricted to functions which pertain to initial database definition and not incremental changes. The grammar of each DBMS this translator accepts is listed in Appendices B and C.

The output of the translator is a source text of NDDL. The output source will be syntactically correct but may be semantically incomplete. Missing items will be flagged in the generated source by comments. Such things as the NTM host must be supplied by the user. The form interface to the NDDL processor (another CPCI) allows the user stepwise refinement of the output NDDL. The grammar for the output of the translator is given in Appendix A.

3.2 Detailed Functional Requirements

The Translator function is to accept and process native DDL subsets which have NDDL equivalents.

3.2.1 Translating DDL Subsets

The Translator accepts the subset of the native DDL which pertains to initial database definition. While all of the subset is accepted only entities within the subset which have NDDL equivalents are translated (e.g.; defining a file of a database is translated but the type of disk drive is accepted and ignored).

3.2.1.1 Inputs

The translator checks the syntax of the input but assumes the semantics are correct for the appropriate DDL. The grammar accepted by the translator is given in Appendix A.

3.2.1.2 Processing

DDL subset statements are read in and recognized and an internal data structure is created and filled in with data. This data structure is common for all DDLs and supports a common NDDL output writer. The structure is illustrated in Figure 3-2. In the following picture a data structure is represented by a box with the name of the data structure above it. The arrows represent pointers to data structures. An arrow pointing to nothing indicates a pointer to another instance of the data structure.

```
       database
      +-----------+
      | nxtdb     |
      +-----------+
      | db_name   |
      +-----------+
      | db_type   |
      +-----------+
      | db_host   |
      +-----------+
      | pass_word |      file_list
      +-----------+     +-----------+
      | file_list | ->  | nxtfile   | ->
      +-----------+     +-----------+
                        | file_name |
                        +-----------+

       table
      +-------------+
      |   nxttab    |   ->
      +-------------+
      | table_name  |
      +-------------+
      | db_name     |
      +-------------+
      | column_list |--> column_list
      +-------------+   +-------------+
                        | nxtcol      | ->
                        +-------------+
                        | level_no    |
                        +-------------+
                        | column_name |
                        +-------------+
                        | union_type  |
                        +-------------+
                       /               \
                      /                 \
+---------------------+                   +---------------------+
|                                                               |
+-------------+  +-------------+  +---------------+ +-----------+
| data_type   |  | redefined_  |  | unknown_type  | |filler_size|
+-------------+  | column_name |  +---------------+ +-----------+
| key         |  +-------------+
+-------------+
```

Figure 3-2. NDDL Data Structure

```
 sets
+-------------+
| nxtset      | ->
+-------------+
| set_name    |
+-------------+
| table_name  |
+-------------+
| table_list  | -> table_list
+-------------+   +-------------+
| link_name   |   | nxttab      | ->
+-------------+   +-------------+
                  | table_name  |
                  +-------------+

 describe
+--------------+
| nxtdesc      | ->
+--------------+
| desc_type    |
+--------------+
| desc_object  |
+--------------+
| desc_name    |
+--------------+
```

Figure 3-2. NDDL Data Structure (Continued)

3.2.1.3 Outputs

The output of the translator is a subset of the NDDL which pertains to the definition of an IS for the CDM. The grammar for the output subset of NDDL is given in Appendix A. The output NDDL will be syntactically correct but may require user modification to be semantically acceptable to the NDDL processor. The translator will supply dummy default values for items not furnished by the DDL and notify the user of them.

After translation the CDMA may refine the generated NDDL. This could be performed using the interactive or batch mode NDDL processor and the translator issued modification requirements as a guide. The modification requirements are in the form of comments embedded in the generated NDDL. They immediately follow the entry to be modified. Their appearance is as follows:

```
/* MOD - message text */
```

3.3 Performance Requirements

3.3.1 Programming Methods

The DDL translator will require a parser. It is intended that a parser generator (namely YACC) be used to create a parser if the grammar is sufficiently complex (e.g. SQL). The interactive NDDL processor is used to refine the generated NDDL.

3.3.2 Program Organization

The program organization of the NDDL translator is subject to change as the design is refined. In general each native DDL will require a separate lexical analyzer and parser with some semantic procedures which fill out data structures as described in section 3.2. A code generator writes out NDDL from these data structures.

3.3.3 Modification Consideration

The NDDL translator makes use of languages which are documented in [4], [6], and [7] in Section 2. Modifications to any of these with respect to their database definition capabilities would require the modification of the translator.

3 4 Human Performance

Human performance requirements are not applicable to this computer program.

3.5 Data Base Requirements

Data base requirements are not required.

3.6 Adaptation Requirements

None.

3.7 Government-Furnished Property List

None.

SECTION 4

QUALITY ASSURANCE PROVISIONS

4.1 Introduction and Definition

"Testing" is a systematic process that may be preplanned and explicitly stated. Test techniques and procedures may be defined in advance and a sequence of test steps may be specified. "Debugging" is the process of isolation and correction of the cause of an error.

"Antibugging" is defined as the philosophy of writing programs in such a way as to make bugs less likely to occur and when they do occur, to make them more noticeable to the programmer and the user. In other words, as much error checking as is practical and possible in each routine should be performed.

4.2 Computer Programing Test and Evaluation

The quality assurance provisions for test will consist of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests will be performed by the design team. Structured design, design walk-through and the incorporation of "antibugging" facilitate this testing by exposing and addressing problem areas before they become coded "bugs".

Tests will consist of inputting an example of each native DDL statement (DATABASE 2 as documented in appendix B and TOTAL DDL as documented in appendix C) to the translator and verifying that the output is correct by comparing it with the NDDL as documented in Appendix A. The generated NDDL will be submitted to the NDDL processor to check its syntax and semantics.

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software will be the Integrated Information Support System (IISS) Test Bed site. The software associated with each CPCI release will be delivered on a media which is compatible with the IISS Test Bed. The release will be clearly identified and will include instructions on procedures to be followed for installation of the release. Integration with the other IISS CPCI's will be done on the IISS TEST BED on a scheduled basis.

SECTION 6

NOTES

Please refer to the Software Availability Bulletin, Volume III, Part 16, CI# SAB620326000, for current IISS software and documentation availability.

APPENDIX A

NDDL GRAMMAR

These are the NDDL commands that may be generated by the translator. Each command is listed for DATABASE 2 and TOTAL. Version 2.3 of NDDL is used. In the following a word that begins with a capital letter is a keyword and must be typed in exactly as it appears. An all lower case word is a name supplied by the user. A token which appears in braces { } indicates one must be selected. A token which appears in brackets [] indicates the token is optional. A token which is followed by an elipsis indicates the token may be repeated.

Define Database

DB2

```
- Define DB2 Database Named db_name On Host host_name;
```

Note: A modification requirement will be issued for the user to supply a host name.

TOTAL

```
- Define TOTAL Database Named db_name On Host host_name
  With Files file_name ... ;
```

Note: A modification requirement will be issued for the user to supply a host name.

Define Table

DB2

```
- Define Table table_id
  With Columns
     { column_name Datatype data_type_name } ... ;
```

Note: SQL data types will be translated as follows:

SQL	NDDL
INTEGER	INTEGER
SMALLINT	SMALLINT
FLOAT	FLOAT
DECIMAL(n,m)	DECIMALnnn_mmm
CHAR(n)	CHARnnn
VARCHAR(n)	VARCHARnnn
LONGVARCHAR	LONGVARCHAR

TOTAL

```
- Define Table table_id
  With Columns
      / [level_no] column_name_1                      \
      |     / Datatype data_type_name [Unique Key] \  |
      |    <  Redefines column_name_2               > |
     <      \ Unknown                              /   >
      |                                               |
      | [level_no] Filler filler_size                 |
      \                                               /

     ... ;
```

Note: A modification requirement will be issued for each filler field to use a named field.

Define Set

DB2

- Not Applicable

TOTAL

```
- Define Set set_name
  From table_id1 to table_id2 ...
  Linked By column_name;
```

Note: column_name is from table_id1.

Describe

```
- Describe Comment_On Of
     / Table Class table_id     \
     \ Column Class column_name /
     "string"
      ;
```

APPENDIX B

DATABASE 2 SYNTAX

The following are DATABASE 2 commands that could appear in a database definition. In the following a word that begins with a capital letter is a keyword and must be typed in exactly as it appears. An all lower case word is a name supplied by the user. An entity which appears in braces { } indicates one must be selected. An entity which appears in brackets [] indicates the entity is optional. An entity which is followed by an elipsis indicates the entity may be repeated. Note that several databases and tables may be defined in one source file.

```
Create Database (maps to NDDL Define Database)
     Create Database db_name
          ...

Create Table (maps to NDDL Define Table)
     Create Table table_id
          ( { column_name data_type [ Not Null ] } ,... )
             / In [ db_name. ] tablespace_name \
             \ In Database db_name             /
          ...

Comment On (maps to NDDL Describe)
     Comment On / Table table_id              \
                \ Column table_id.column_name /
        "string"

Commands that may appear in DDL.
     Create Index
     Create Stogroup
     Create Synonym
     Create Tablespace
     Create View
     Grant (privileges)
```

APPENDIX C

TOTAL DDL SYNTAX

The following are statements from the TOTAL DDL which could appear in a database definition. TOTAL is a fixed format line oriented language and could be parsed by little more than a lexical analyzer. Statements which map to NDDL are prefixed by a star. In the following a word that begins with a capital letter is a keyword and must be typed in exactly as it appears. An all lower case word is a name supplied by the user. A name of mmmm indicates a master table name. A name of vvvv indicates a variable table name. A token which appears in braces { } indicates one must be selected. A token which appears in brackets [] indicates the token is optional. A token which is followed by an elipsis indicates the token may be repeated. The notation ".pp." indicates a level number for a column (e.g. a COBOL record).

Note that one database and several master and variable tables may be defined in one source file. All TOTAL data fields will be of the data type CHAR_n (where n is the size in bytes of the field. A warning will be issued for the user to modify these fields. The data definition should not contain fillers, use the database definition which has all fields named. In variable records the RECORD CODE redefines the last data field in the BASE DATA section. In order to facilitate the redefinition the level number of a RECORD CODE field is one and that of following fields will be incremented by two. Link and key fields which have secondary names will have the secondary name treated as a redefinition of the field.

```
        BEGIN-DATA-BASE-GENERATION:
        OPTIONS: ...
        LOGGING: ...
        CTLX: ...
*       DATA-BASE-NAME = xxxxxxxx ; Maps to NDDL Define Database
        SHARE-IO: ...
        IOAREA = ...
        JCL = ...

        BEGIN-MASTER-DATA-SET:
*       DATA-SET-NAME = mmmm       ; File of Define Database
        IOAREA = xxxx

                                  ; Maps to Define Table
        MASTER-DATA:
*       [.pp.] mmmmROOT = 8       ; named data field
*       [.pp.] mmmmCTRL = n        ; named data, key field
*       [[.pp.] mmmmLKxx = 8]      ; named data, link field
*       [[.pp.] xxxxxxxx [= n]]    ; named data or unknown field
*       [[.pp.] *FILLER* = n]      ; unnamed data field
        END-DATA:
```

```
                                 ; Physical data description
    DEVICE = ...
    ACCESS-METHOD = ...
    TOTAL- LOGICAL-RECORD = ...
    TOTAL-TRACKS = ...

    LOGICAL-RECORD-LENGTH = ...
    LOGICAL-BLOCKS-PER-TRACK = ...
    CONTROL-INTERVAL = ...
    CONTROL-INTERVAL-SIZE = ...
    DISK-EXTENTS = ...
    DOS = ...
    OLD-FILE = ...
    END-MASTER-DATA-SET:

    BEGIN-VARIABLE-ENTRY-DATA-SET:
*   DATA-SET-NAME = vvvv        ; File of Define Database
    IOAREA = ...

                                 ; Maps to Define Table
    BASE-DATA:
*   [[.pp.] vvvvCODE = 2]       ; named data field
*   [.pp.] xxxxxxxx = n [=mmmmCTRL]
                                 ; named data field associated
                                   with key
*   [[.pp.] *FILLER* = n]       ; unnamed data field
*   [.pp.]mmmmLKxx = 8 [=xxxxxxxx]
                                 ; named data, link field of
                                   master record
*   RECORD-CODE = xx             ; named redefines  field
*   [.pp.] xxxxxxxx = n [=mmmmCTRL]
                                 ; named data field associated
                                   with key ? is it unique?

*   [[.pp.] *FILLER* = n]       ; unnamed data field
*   [.pp.]mmmmLKxx = 8 [=xxxxxxxx]
                                 ; link field of master record
    END-DATA:
                                 ; Physical data description

    DEVICE = ...
    ACCESS-METHOD = ...
    TOTAL- LOGICAL-RECORD = ...
    TOTAL-TRACKS = ...
    LOGICAL-RECORD-LENGTH = ...
    LOGICAL-BLOCKS-PER-TRACK = ...
    CONTROL-INTERVAL = ...
    CONTROL-INTERVAL-SIZE = ...
    DISK-EXTENTS = ...
    DOS = ...
    OLD-FILE = ...
    END-VARIABLE-ENTRY-DATA-SET:

    END-DATA-BASE-GENERATION:
```

☆U.S. GOVERNMENT PRINTING OFFICE: 1992 - 648-127/62341